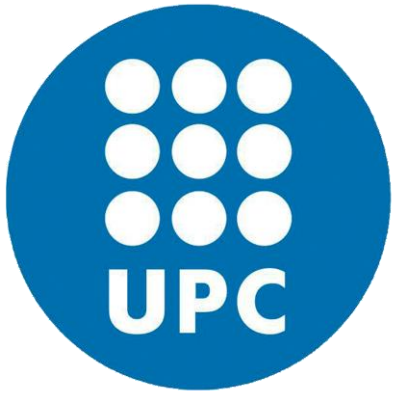


UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

ARTIFICIAL INTELLIGENCE APPLIED TO ELECTROMECHANICAL MONITORING, A PERFORMANCE ANALYSIS

Erasmus project

Authors: Staš Osterc

Mentor: Dr. Miguel Delgado Prieto, Dr. Francisco Arellano Espitia

1/5/2020

ANNEX VI – DECLARACIÓ D'HONOR

I declare that,

the work in this Master Thesis / Degree Thesis (*choose one*) is completely my own work,

no part of this Master Thesis / Degree Thesis (*choose one*) is taken from other people's work without giving them credit,

all references have been clearly cited,

I'm authorised to make use of the company's / research group (*choose one*) related information I'm providing in this document (*select when it applies*).

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary actions by *The Universitat Politècnica de Catalunya - BarcelonaTECH*.

_____	_____	_____
Student Name	Signature	Date

Title of the Thesis : _____

_____

_____

_____

Contents

Table of Figures

Introduction

Abstract

Artificial intelligence is a wide concept and it's being used in more and more machine applications, teaching them to perform tasks which would require human intelligence. The implemented algorithm requires samples of ''experience'' from which it can learn and predict the outcome; from there it mainly feeds itself. AI is basically divided into two subsets; deep learning and machine learning. They are mostly distinguished with the way the data is presented to the network.

To summarize this project; First the data was monitored on an electromechanical system, monitored data from vibrations was saved and brought in Matlab program, there the data was transformed for future use with autoencoders, which learned the conditions, after training we have to try different parameters for getting the closest reconstruction possible.

In the project we applied several techniques like using a multilayered auto-encoder and then finding the best hyper parameters for best results (they were measured by plotting signals and mean square error).

Aim

The purpose of an auto-encoder is to learn a representation for a set of data, by training itself with it. With artificial intelligence one can develop machines that are able to read and understand, to us known as natural learning process.

In our case the aim was to build 3 auto-encoders with multiple hyper parameters and train an auto-encoder in order to learn a representation for a set of data. To achieve that auto-encoder actually learns something from the process, dimensions of the input and the representation can be adjusted.

We learned to construct an artificial neural network for a specific intention of predicting errors in the electromechanical device.

Scope of the work

Our work contains study of machine learning and study of the data for predicting outputs. Our data was extracted from electromechanical scheme from stator current, speed, gearbox vibration and motor vibrations. One step is the design of the corresponding scripts for testing size, neurons, epochs, L2 weight regularization and sparsity proportion and regularization with the included data. At first the values used were random since we were not yet familiar with function of parameters. Through study and analyze of the parameters, they were adjusted. Plotting the signals also gave us some information of parameters functions and how they apply in the autoencoder.

The purpose of our work was to train the machine to recognize the faults of its mechanical components. We had to train the auto-encoder to recognize the faults in the future of an operating machine.

Requirements

As artificial intelligence field is growing each day, there are many platforms to use to get the desired results. Some of more popular are; Google AI platform, Tensor Flow, Microsoft Azure, etc…

The signal processing community has adopted MATLAB as a flexible modelling, simulating and testing software development environment. MATLAB includes numerous toolboxes, open-source code and pre-compiled libraries, which facilitate the design of complex systems using high-level models and provides the means for rapid verification of signal processing algorithms and systems in a user-controlled environment.

Development

Background

Artificial intelligence, is intelligence demonstrated by machines in opposite to natural intelligence displayed by humans. The term artificial intelligence is often used to describe machines that mimic cognitive function which humans associate with the human mind. If we want to understand how artificial intelligence works we first need to understand how human brains work.

Human brain is made of approximately 100 billion nerve cells called neurons. Neurons have the amazing ability to gather and transmit electrochemical signals. Neurons share the same characteristics and have the same makeup as other cells, but the electrochemical aspect lets them to transmit signals over long distances (up to a few meters) and send messages to each other. Which makes our brain perform incredible tasks like: controlling body temperature blood pressure heart rate and breathing, it accepts information about the world around you from various senses (seeing, hearing, tasting, smelling and touching). It handles your physical movement and lets you think, dream, reasons and experience emotions.

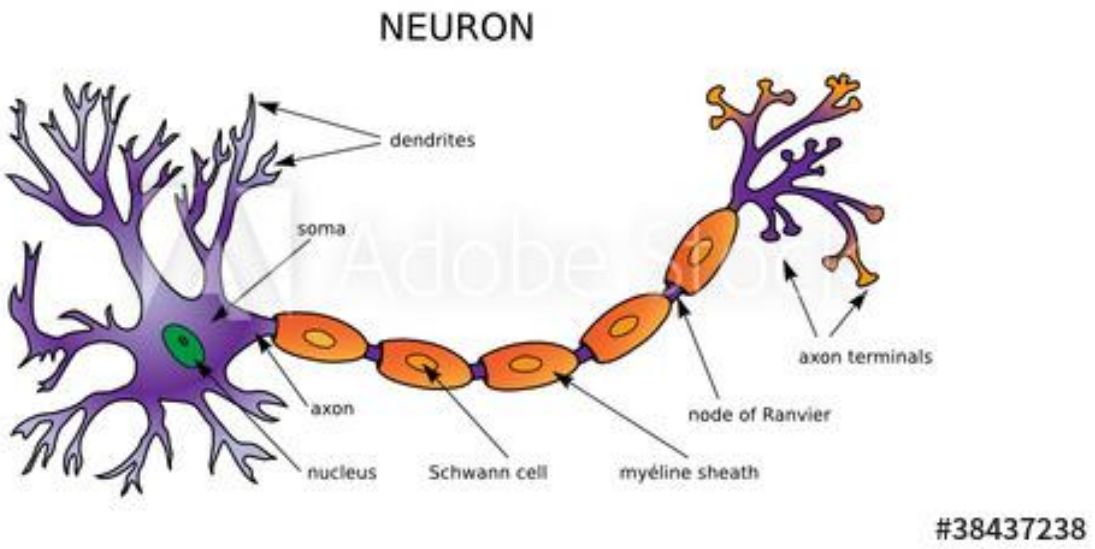


Figure 1: Neuron

If we look closer AI is working on very similar process as human brain. The successes of AI owe much to the arrival of more powerful processors and ever-growing quantities of training data. But the concept that underlies these advances is the artificial neural network. These networks consist of layers of nodes that are analogous to neurons. Nodes in the input layer are connected to nodes in a hidden layer by a series of mathematical weights that act like the synapses between neurons. The hidden layer is similarly connected to an output layer. Input data for a task such as facial recognition could be an array of numbers that describe each pixel in an image of a face in terms of where it falls on a 100-point scale from white to black, or whether it is red, green or blue. Data are fed in, the hidden layer then multiplies those values by the weights of the connections, and an answer comes out. To train the system to produce the correct answer, this output is compared with what it should have been if the output were an exact match for the input, and the difference is used to adjust the weights between the nodes. A more complex version of this process, called a deep neural network, has many hidden layers.

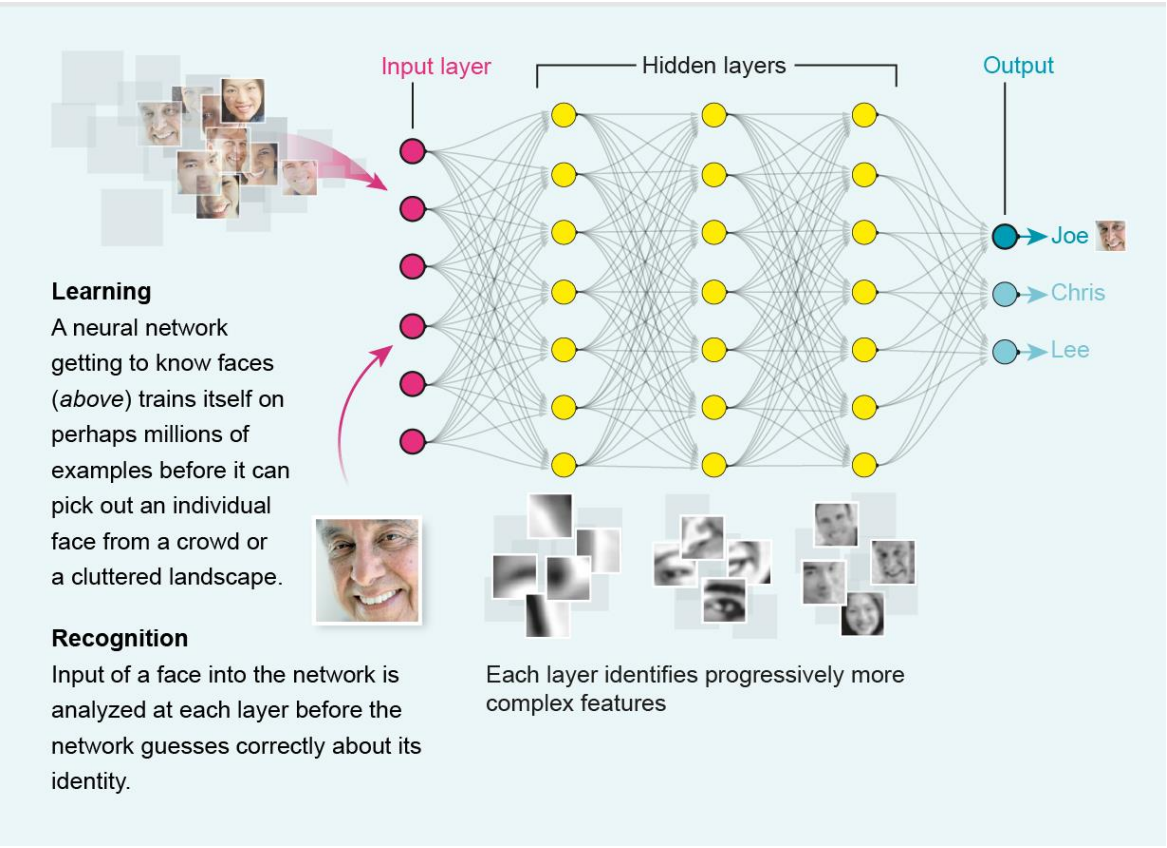


Figure 2: Visual representation of hidden layers

While no one has identified all the **branches of Artificial Intelligence** here is a list of few of them:

- Natural language processing: The branch of AI that gives computers the ability to process, analyse and manipulate human communication, both speech and text. NLP is best represented by devices such as Alexa and Siri.

- Cognitive computing: A cognitive system mimics the human brain and helps improve decision-making by extracting information from unstructured data. Self-learning algorithms, pattern recognition and natural language processing are key elements of cognitive analysis.

- Computer vision: The field of AI that relies on pattern recognition and deep learning to interpret the visual world. It helps computers to analyse images, videos and multi-dimensional data in real time.

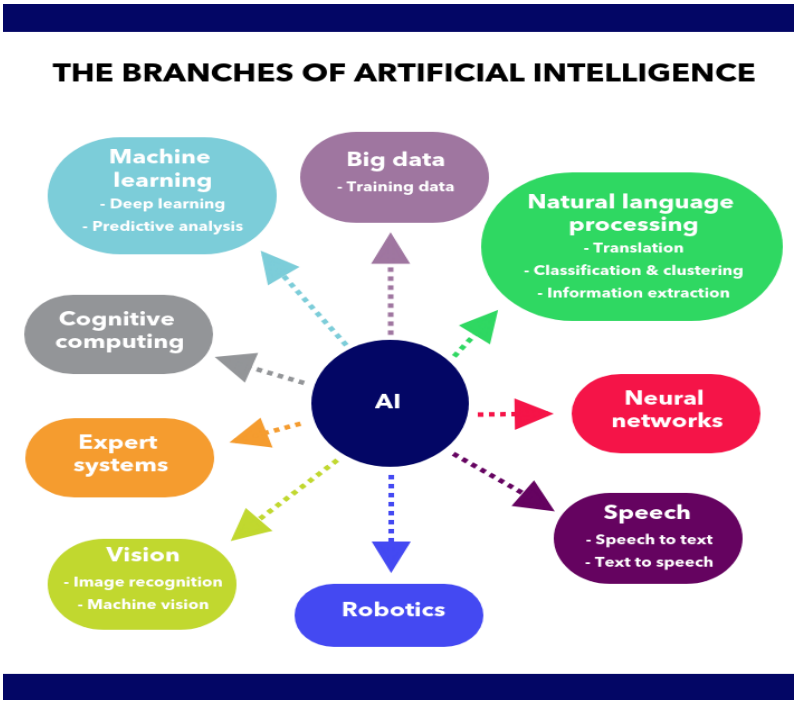


Figure 3 Branches of AI

In short I would say AI is, a systems ability to correctly interpret external data, to learn from such data and to use learning to achieve specific goals and tasks trough flexible adaptation.

State of art review

Artificial Intelligence is a booming field of scientific discovery and practical deployments. Once a mostly academic area of study, twenty-first century AI enables a spectrum of mainstream technologies that are having a substantial impact on everyday lives. In many cases, already now, AI accompanies the users in our everyday errands and professional lives. In the future it will not only reshape business, public administration, health care, finances or education, but may also contribute to solving grand civilizational challenges such as climate change, hunger or inequality. The phase of AI massively transforming society, economy, and politics has already begun.

Machine Learning is an application of artificial intelligence that provides systems to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on development of programs that can get data and use it to learn for themselves.

The process of learning begins with observations of data such as examples, direct experience or instructions in order to look for patterns in data and make better decisions in the future based on examples that we provide. The primary aim is to allow computers to learn automatically without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms are often categorized as supervised or unsupervised:

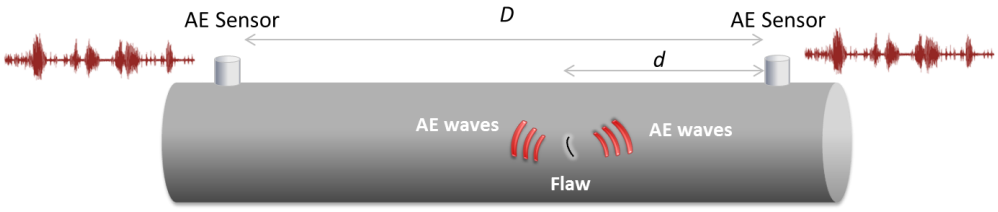
- Supervised machine learning algorithms can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from analysis of known training dataset, the learning algorithm produces inferred function to make predictions about output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- Unsupervised machine learning algorithms when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. The system doesn't figure out the right output, but it explores the data and can draw from interferences from datasets to describe hidden structures from unlabelled data.

Deep learning is a type of machine learning in which a model learns to take classification tasks directly from images, sound or text. Deep learning is usually implemented using a neural network architecture. The form "deep" refers to numbers of layers in the network-the more layers the deeper the network. Traditional neural networks contain only 2 or 3 layers, while deep neural networks can have hundreds.

The subject of machine condition monitoring and fault diagnosis as a part of system maintenance has gained a lot of interest due to potential benefits to be learnt from reduced maintenance budgets, enhanced productivity and improved machine availability. Artificial intelligence is successful method of machine condition monitoring and fault diagnosis since these techniques are used as tools for routine maintenance. Intelligent system such as artificial neural network (ANN), fuzzy logic system (FLS), genetic algorithms (GA) have developed many different methods. However the use of acoustic emission (AE) signal analysis and AI techniques for machine condition monitoring is still rare. (Ali, 2018)

In **acoustic emission analysis**, the waves are sent from an emission source and transferred to the surface by transmission medium. The low displacement or high-frequency mechanical waves can be

picked up as electronic signals. The signal strength can be increased using preamplifier before the data are interpreted by AE equipment.



$$d = \frac{1}{2}\left(D - \Delta T \cdot V\right)$$

d = distance from first hit sensor

D = distance between sensors

V = wave velocity

Figure 1 Acoustic emission sensor

Acoustic Emission (AE) analysis is one of the most effective monitoring methods with high sensitivity and reliability in detection of changes in materials. Consequently, AE has been used successfully in many engineering applications and for a broad variety of materials, material compositions and structures. The engineering applications include tribology, failure of components and, more recently, laser welding and/or additive manufacturing. Full example of work performed can be seen here (WASMER, 2019).

Approach

First step was getting familiar with branches of artificial intelligence. We did some research about unsupervised and supervised learning based on the data and information we had. Unsupervised learning searches for hidden patterns or structures in the data with clustering (k-Means, k-Medoids, hierarchical clustering, …). Clustering gathers data into groups based on the characteristics. On the other hand, supervised learning works with labelled data and the output pattern is provided to the system.

Information flow through neural network can be divided in two ways; when the network is learning and when network is already trained. We fed the data to the network via input units.

The simplest approach was guessing the parameters, which gave us a rough evaluation with the output. After getting an estimate, we tried with recommended values, repeat it many times and keep track with the results. The analyzing of parameters was delivered with the function, which for each value runs the program 10 times and collects the results. Choosing the best parameters was based on the lowest random error of the trained network. Before analyzing, the data was split in test set and train set to avoid underlying noise.

Design and Methodology

To simplify some sections we used shortcuts like loops and functions to fasten the obtaining of results.

To understand and to properly approach the task, we studied key points of how to group the data and train a neural network, from which we would gain a visual representation of the reconstruction.

Data is reduced to ''chunks'' which are delivered to the network in smaller dimensions for undisturbed encoding. Therefore, with the number of samples increasing, algorithms improve their performance.

With stacking the auto-encoders we provided noise to input which passes through hidden layers. The data collected from the first neural network (output) was later used as a new input in the second neural network and so on to the third neural network. Also, the error was calculated between output and input. The last output collected was our new data.

We continued this procedure until the reconstruction was reduced to two dimensions to minimize the error. Analyzing the auto-encoders parameters also helped reduce the error and optimized the reconstruction.

Stage one: Monitoring signal

While we are working with electromechanical system the first step is to determine which monitoring technique we are going to use for the purpose of fault detection.

For our project we used vibrations monitoring;

Vibration analysis is a process of looking for anomalies and monitoring change from the established vibration signature of a system. The vibration of any object in motion is characterized by variations of amplitude, intensity, and frequency. These vibration signatures can be correlated to physical phenomena, making it possible to use vibration data to gain insights into the health of equipment.

Vibration data is captured by accelerometers installed on one or more orthogonal axes. The sampling rate of the accelerometers needs to be fast enough to capture the behaviour of interest. Next, the signal needs to be digitized at an appropriate sampling rate to enable it to be digitally reconstructed. The result is the time waveform (oscillation amplitude as a function of time) of vibration along the axes of interest.

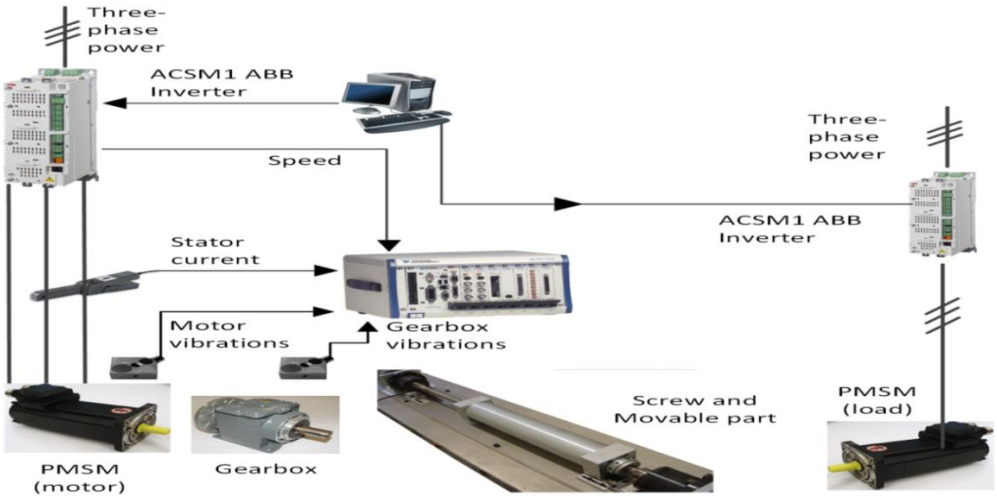


Figure 2 Test bench

Stage two: Signal pre-processing

If there is much irrelevant information present or noisy and unreliable data, then knowledge discovery during the training phase can be more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data pre-processing includes; cleaning, normalization, instance selection, transformation, feature extraction and selection, etc…

The next step is to perform a fast-Fourier transform (FFT) algorithm to convert the time waveform into a vibration frequency spectrum. The scope of the frequency spectrum depends upon the accelerometers and the analog-to-digital converter (ADC) used. Much of the insight delivered by vibration analysis is based upon correlating frequency spikes to physical characteristics of the system.

Stage three: Deriving Features

Deriving futures is one of the most important parts of machine learning, it basically turns raw data into information that a machine learning algorithm can use.

Techniques for feature section tasks of signal data:

- Peak analysis (preform an FFT and identify dominant frequencies)
- Pulse and transition metrics (derive signal characteristics such as rise time, fall time and setting time)
- Spectral measurements (plot signal power, bandwidth, mean frequency and median frequency)

Stage four: Build and train model

When we pre-process the signal and derive futures, next step is to build a model and start training it. After selecting the algorithms, we need to train the model where we input the data into the model. A critical step here is model accuracy. While there are no widely accepted or internationalised thresholds, it is vitally important to establish model accuracy within your selection framework. Setting a minimum acceptable threshold and applying a great statistical discipline is key, we have to retrain the model as it is natural the models may need some fine-tuning.

Stage five: Finding the best model

A good model includes only the futures with the most predictive power. A simple model that generalizes well is better than a complex model that may not generalize or train well to new data.

An **autoencoder** is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of autoencoder is to learn a representation (encoding) for a set of data. Along with the reduction side, a reconstruction side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original output.

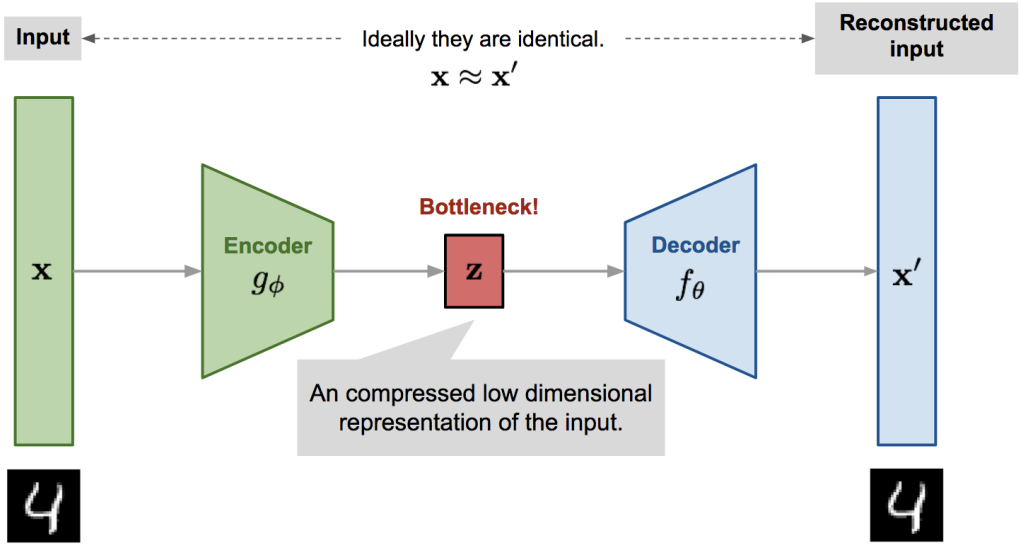


Figure 3 Visual representation of autoencoder

How autoencoder works

Autoencoders are the models in a dataset that find low-dimensional representations by exploiting the extreme non-linearity of neural networks. An autoencoder is made up of two parts:

Encoder part – This transforms the input (high dimensional into a code that is short and crisp)

Decoder part – This transforms the short code into high dimensional output

Let's take example of autoencoder presented in Matlab and define all the hyperparameters:

```
autoencl = trainAutoencoder(TrainData,450,...
    'MaxEpochs',400,...
    'SparsityProportion',0.4,...
    'DecoderTransferFunction','logsig',...
    'SparsityRegularization',1,...
    'L2WeightRegularization',1e-05,...
    'EncoderTransferFunction','logsig');
enc01 = encode(autoencl,TrainData);
```

Max Epochs - An epoch refers to one cycle through the full training dataset. Usually, training a neural network takes more than a few epochs. In other words, if we feed a neural network the training data for more than one epoch in different patterns, we hope for a better generalization when given a new input (test data).

Sparsity Proportion - Desired proportion of training examples a neuron reacts to. Sparsity proportion is a parameter of the sparsity regularizer. It controls the sparsity of the output from the hidden layer. A low value for Sparsity Proportion usually leads to each neuron in the hidden layer "specializing" by only giving a high output for a small number of training examples. Hence, a low sparsity proportion encourages higher degree of sparsity.

Sparsity Regularization - Coefficient that controls the impact of the sparsity regularizer in the cost function.

L 2Weight Regularization – A linear regression model that implements L2 norm of regularization is called ridge regression.

$$||W_2|| = (w_1^2 + w_2^2 + \cdots + w_N^2)^{\frac{1}{2}}$$

To implement this, linear regression model stays the same

$$\hat{\boldsymbol{y}} = \boldsymbol{w_1 x_1} + \boldsymbol{w_2 w_2} + \cdots + \boldsymbol{w_N x_N} + \boldsymbol{b}$$

But it is the calculation of the loss function that includes these regularisation terms:

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

The regularisation terms are 'constraints' by which an optimisation algorithm must 'adhere to' when minimising the loss function, apart from having to minimise the error between the true *y* and the predicted $\hat{y}$.

Encoder Transfer function - Transfer function for the encoder; Logistic Sigmund function.

$$f(z) = \frac{1}{1 + e^{-z}}$$

Decoder transfer function – Transfer function for the encoder; Alse Logistic Sigmund function

$$f(z) = \frac{1}{1 + e^{-z}}$$

When we finish training autoencoder, we can calculate **mean squared error** (MSE). The mean squared error tells you how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the "errors") and squaring them. The squaring is necessary to remove any negative signs. It also gives more weight to larger differences. It's called the mean squared error as you're finding the average of a set of errors. The smaller the mean squared error is the closer you are to finding the line of best fit.

The goal of **reconstruction** is to go from discrete time signal to continuous time signal. Basically we want to choose the smoothest signal that goes through a set of samples.

As we obtained the reconstruction, we can calculate mean squared error from our original and reconstructed data.

```
% Mean squared error
   mseError = mse(TrainData-Xreconstructed);
```

Figure 4 Matlab example of calculating mse

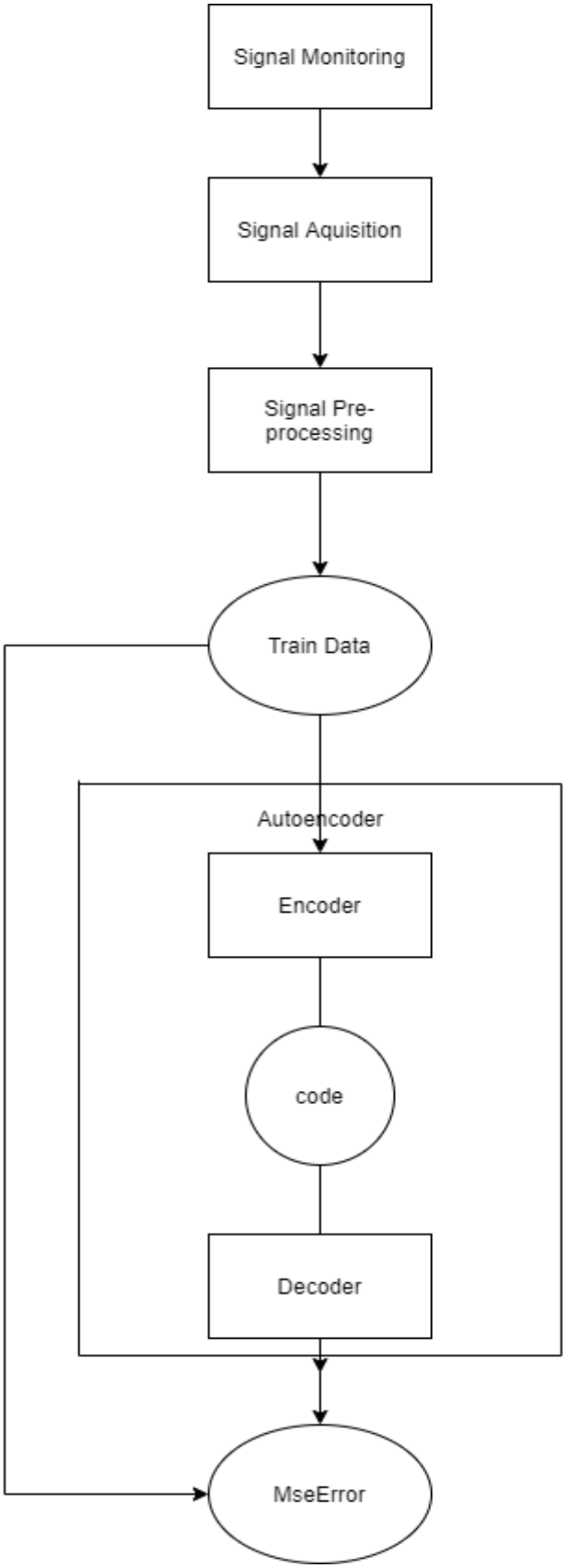


Figure 5 Block diagram for Methodology

Summary of results

Experimental Results

While getting the closest reconstruction is mostly trial and error, we need to test a bunch of parameters in order to find our best solution.

After the data was extracted, features derived then we imported data into Matlab, we get a new signal of dimension (2720000x1) all the conditions are included in that file, but for future use we had to break them down into chunks and create new matrix.

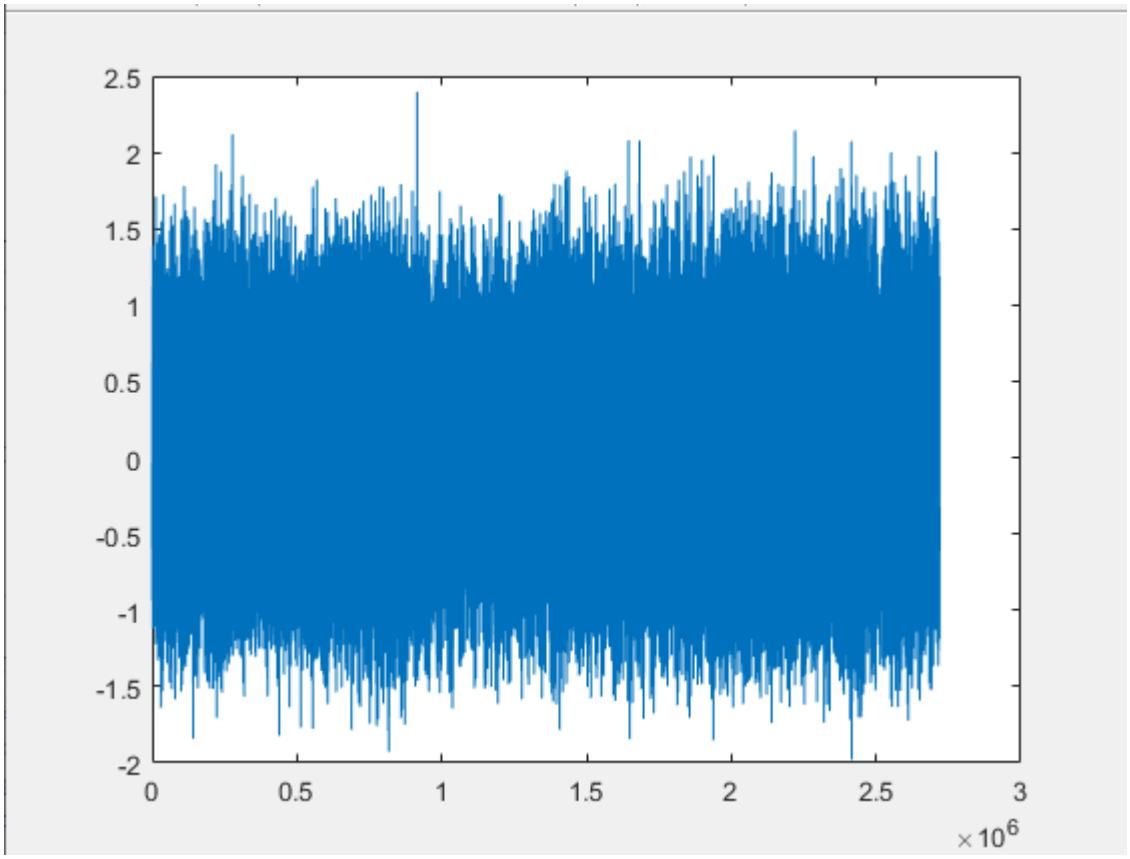


Figure 6 Plot of new Signal

New Matrix of training data has (820x400) data points.

820 is the number of points on X axis and 400 is the number where all conditions are stored in.

So we memorize healthy condition from 1-80, bearing fault condition from 81-160, demagnetized fault condition from 161-240, eccentricity fault condition from 240-320 and gear fault condition from 321-400. With that we created a file that autoencoder can work with.

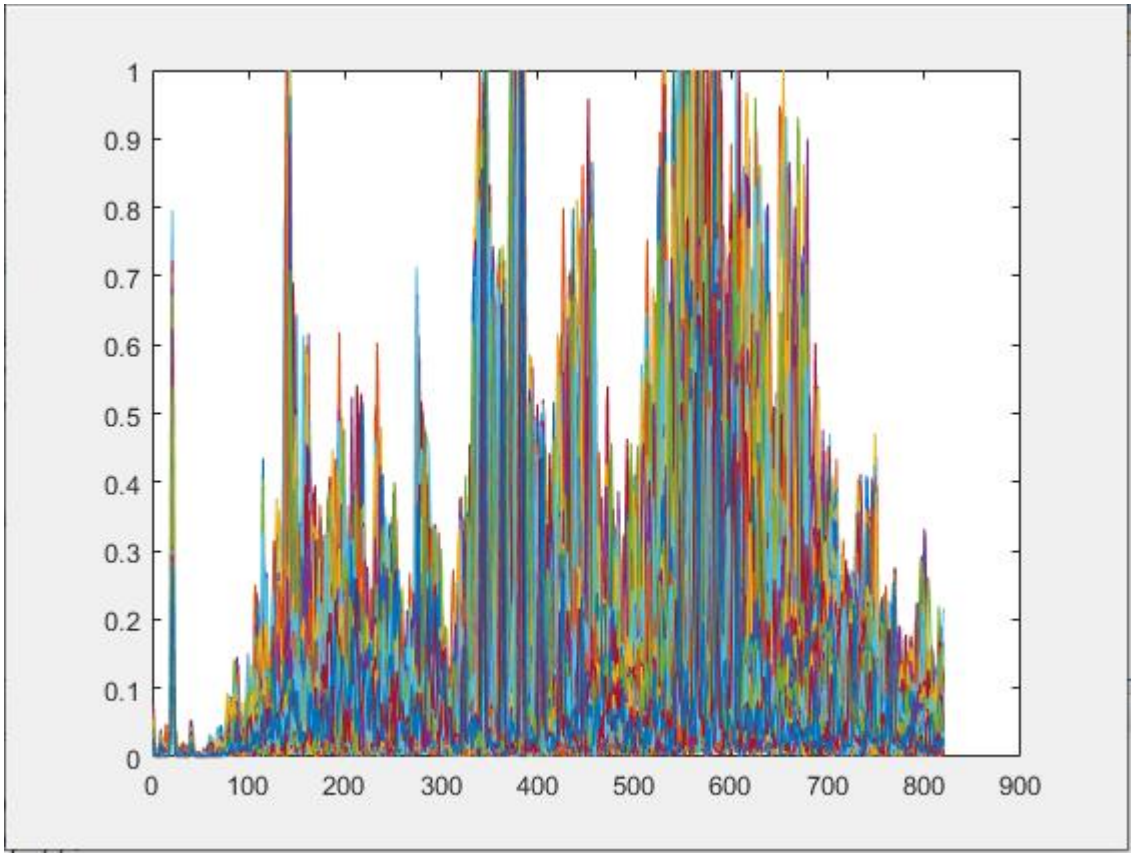


Figure 7 Plot of all conditions in a matrix

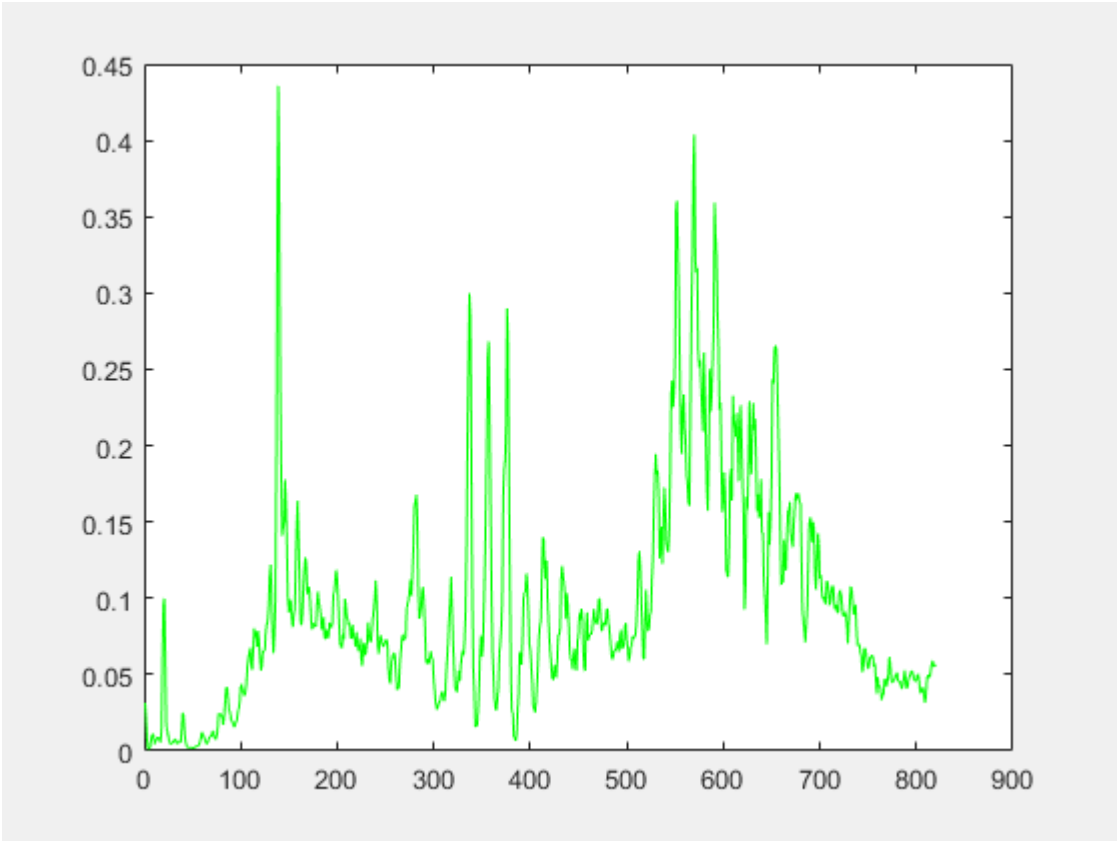


Figure 8 Plot for healthy condition

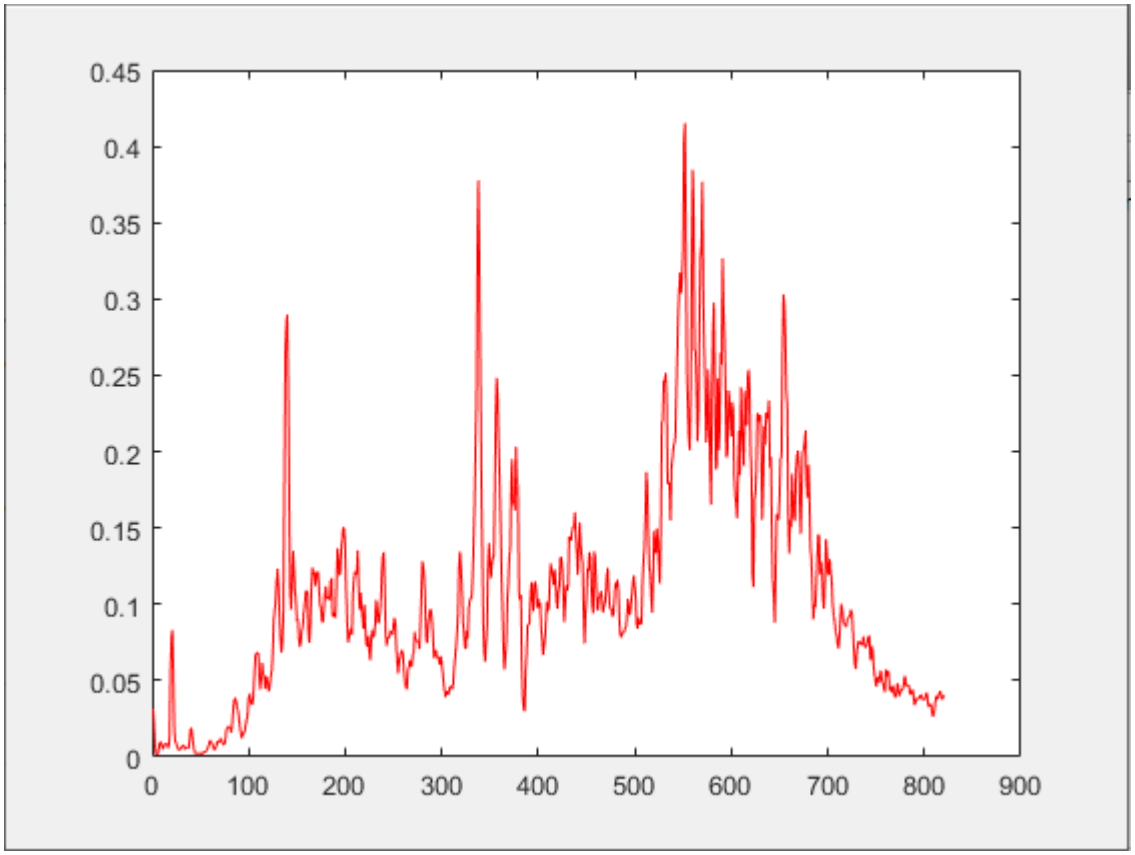


Figure 9 Plot of bearings fault condition

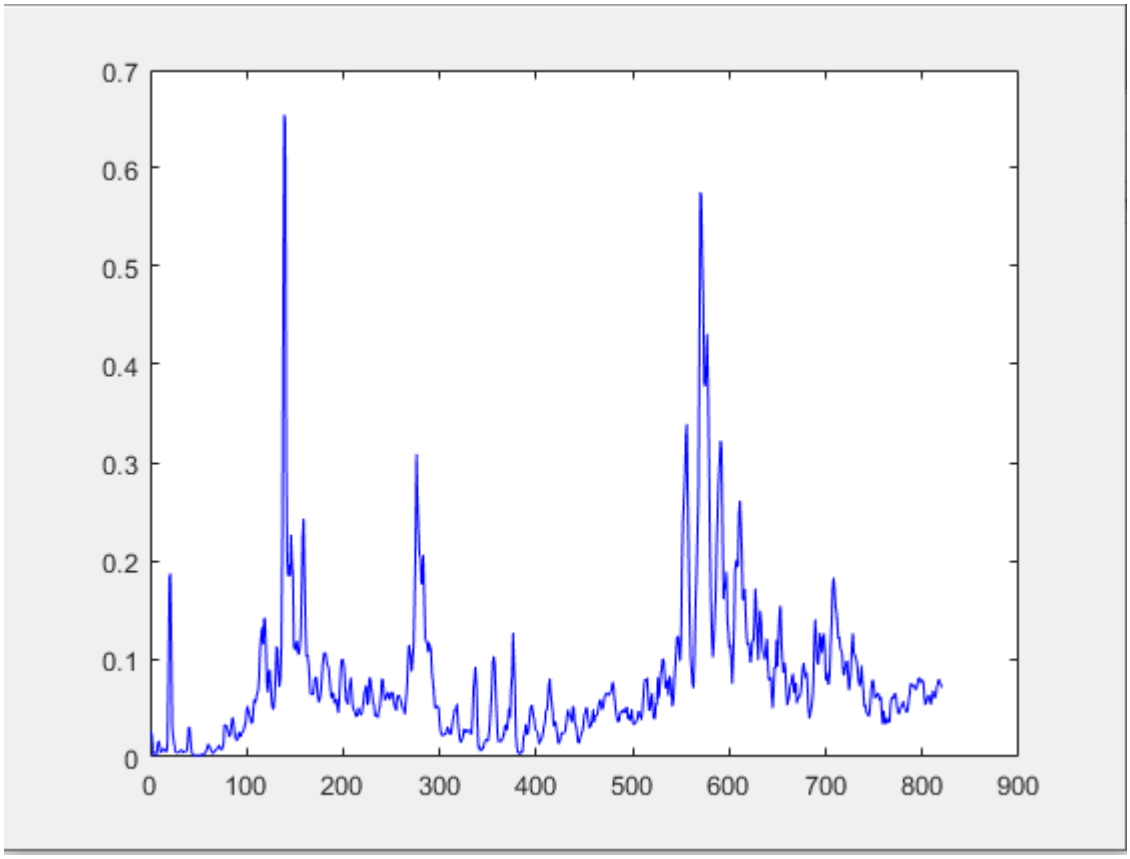


Figure 10 Plot of demagnetization fault condition

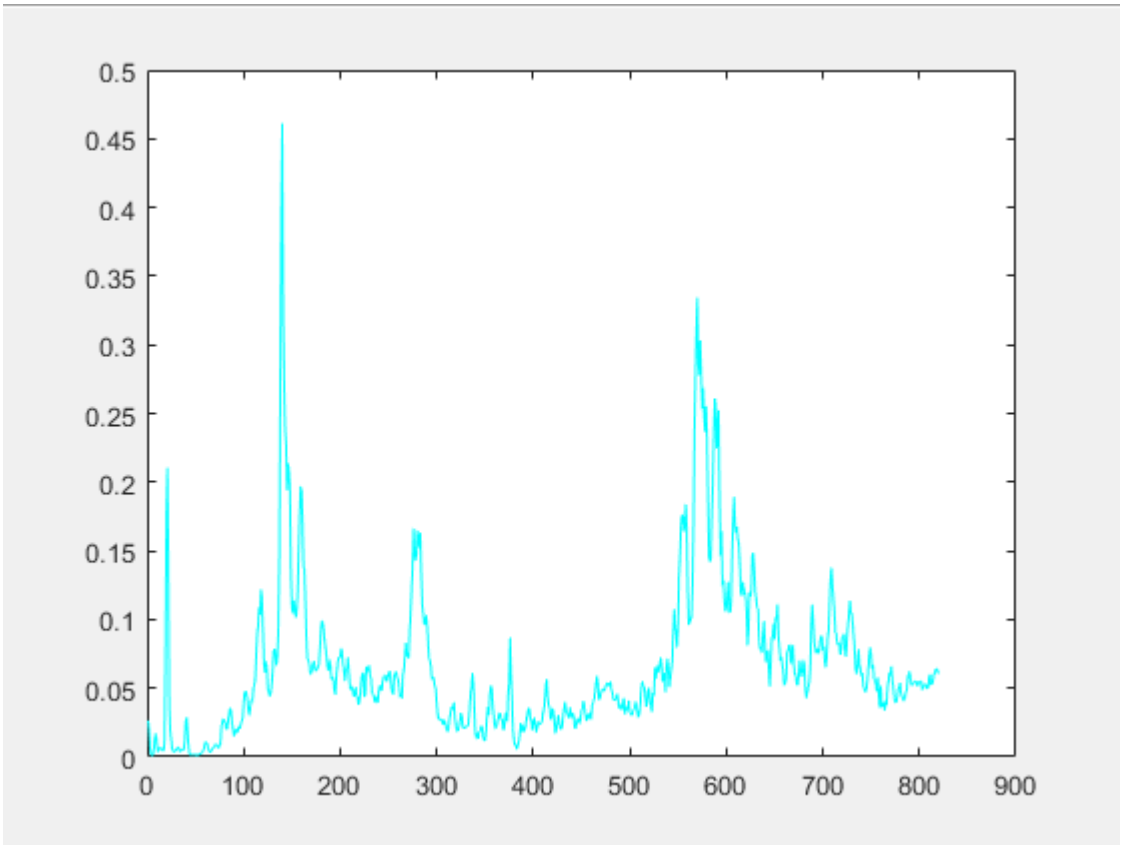


Figure 11 Plot of eccentricity fault

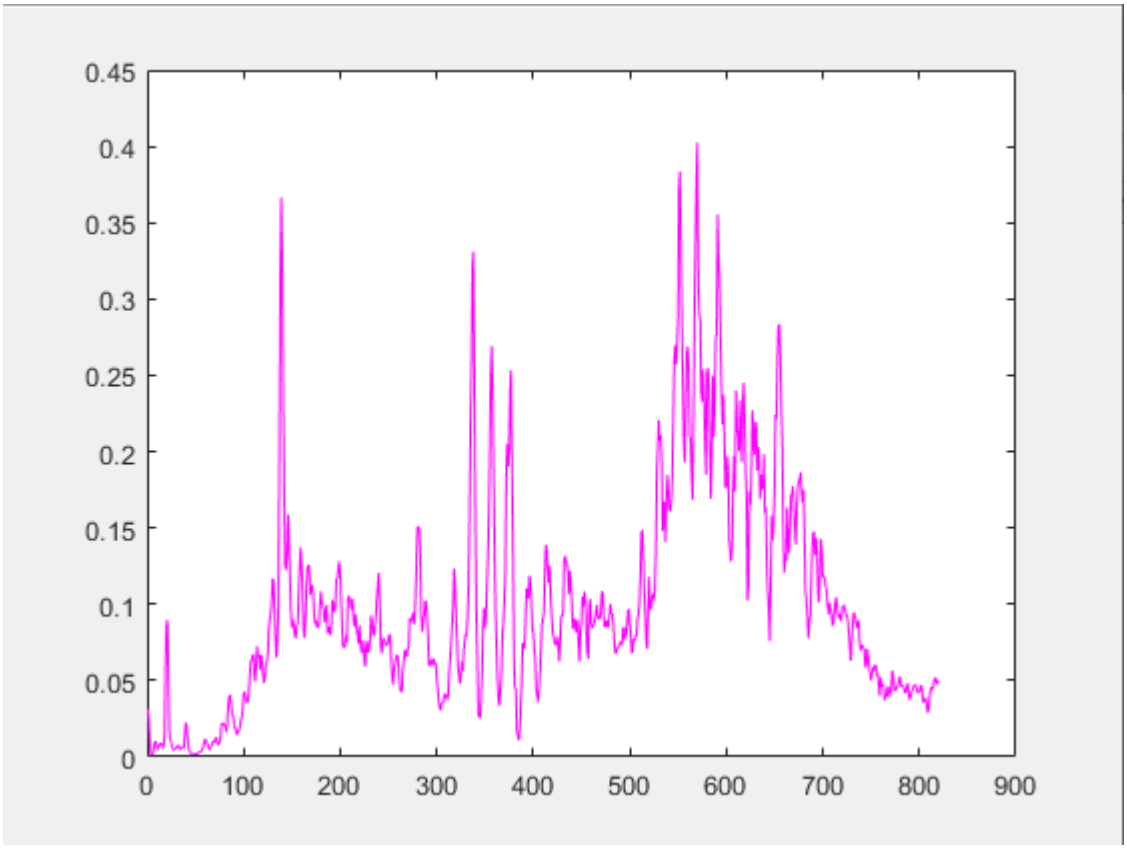


Figure 12 Plot of gear fault condition

Now that we have training data, we can start training our neural network. First we had to figure out what hidden layers of first and second autoencoder had the best performance. For that we used scripts that tested 9 different hidden layers one [200, 250, 300, 350, 400, 450, 500, 550, 600]. Each value was tested 10 times in order to get average MSE error for each condition, other parameters we kept fixed for now.

```
autoenc1 = trainAutoencoder(TrainData,hl1,...
    'MaxEpochs',400,...
    'SparsityProportion',0.4,...
    'DecoderTransferFunction','logsig',...
    'SparsityRegularization',0.0005,...
    'L2WeightRegularization',0.000005,...
    'EncoderTransferFunction','logsig');
enc01 = encode(autoenc1,TrainData);
```

Figure 13 Autoencoder one for testing first hidden layer

As hidden layer one finished training we kept the best result of hidden layer one (450) and moved to training hidden layer two with values [10, 25, 50, 100, 150, 200, 250].

```
autoenc2 = trainAutoencoder(enc01,hl2,...
    'MaxEpochs',400,...
    'SparsityProportion',0.4,...
    'DecoderTransferFunction','logsig',...
    'SparsityRegularization',0.00005,...
    'L2WeightRegularization',0.00005,...
    'EncoderTransferFunction','logsig');
enc02 = encode(autoenc2,enc01);
```

Figure 14 Autoencoder two for testing second hidden layer

For third autoencoder we kept value of hidden fixed to [2] in order to obtain 2D reconstruction later.

When each value gets tested 10 times we obtained the best configuration of each hidden layer. That is the smallest MSE error.

```
fx The best configuration is: 450, 200>>
```

Figure 15 Best configuration for hidden layer one and two

After the first script finished training, we moved to testing Weight Regularization with values [0.1, 0.01, 0.001, 0.0005, 0.0001, 0.00005, 0.00001, 0.000005, 0.000001].

```
autoenc1 = trainAutoencoder(TrainData,450,...
    'MaxEpochs',400,...
    'SparsityProportion',0.4,...
    'DecoderTransferFunction','logsig',...
    'SparsityRegularization',SR,...
    'L2WeightRegularization',L2W,...
    'EncoderTransferFunction','logsig');
enc01 = encode(autoenc1,TrainData);
```

Figure 16 Autoencoder one for testing L2 Weight Regularization

With the best value for L2 Weight Regularization, the testing of Sparsity Regularization began; Values for Sparsity Regularization were [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.01, 1].

The best configuration for L2 Weight Regularization and Sparsity proportion is:

```
TThe best configuration is: 1.000000e-05, 1>>
```

Figure 17 Best configuration for L2 Weight Regularization and Sparsity proportion

The last parameter we tested was Sparsity Regularization, for that we take the values we obtained before and tested Sparsity with values of [0.05, 0.1, 0.2, 0.3, 0.35, 0.04, 0.045, 0.5, 0.55, 0.6, 0.65].

```
autoencl = trainAutoencoder(TrainData,450,...
       'MaxEpochs',400,...
       'SparsityProportion',SP,...
       'DecoderTransferFunction','logsig',...
       'SparsityRegularization',1,...
       'L2WeightRegularization',1.0e-05,...
       'EncoderTransferFunction','logsig');
   enc01 = encode(autoencl,TrainData);
```

Figure 18 Autoencoder for testing Sparsity Regularization

The best configuration for Sparsity Regularization is:

```
TThe best configuration is: 3.000000e-01, >>
```

Figure 19 The best configuration for Sparsity Regularization

Now as we have the best parameters we can build our final model of autoencoders and calculate our MSE error for each condition.

- The input function for **healthy** condition is plotted in "green" and reconstruction of that condition is plotted in black .
- The input function for **bearings fault** condition is plotted in "red" and reconstruction of that condition is plotted in black.
- The input function for **demagnetization fault** condition is plotted in "blue" and reconstruction of that condition is plotted in black.
- The input function for **eccentricity fault** condition is plotted in "cyan" and reconstruction of that condition is plotted in black.
- The input function for **gears fault** condition is plotted in "magenta" and reconstruction of that condition is plotted in black.

Also the MSE error for each condition is figured.

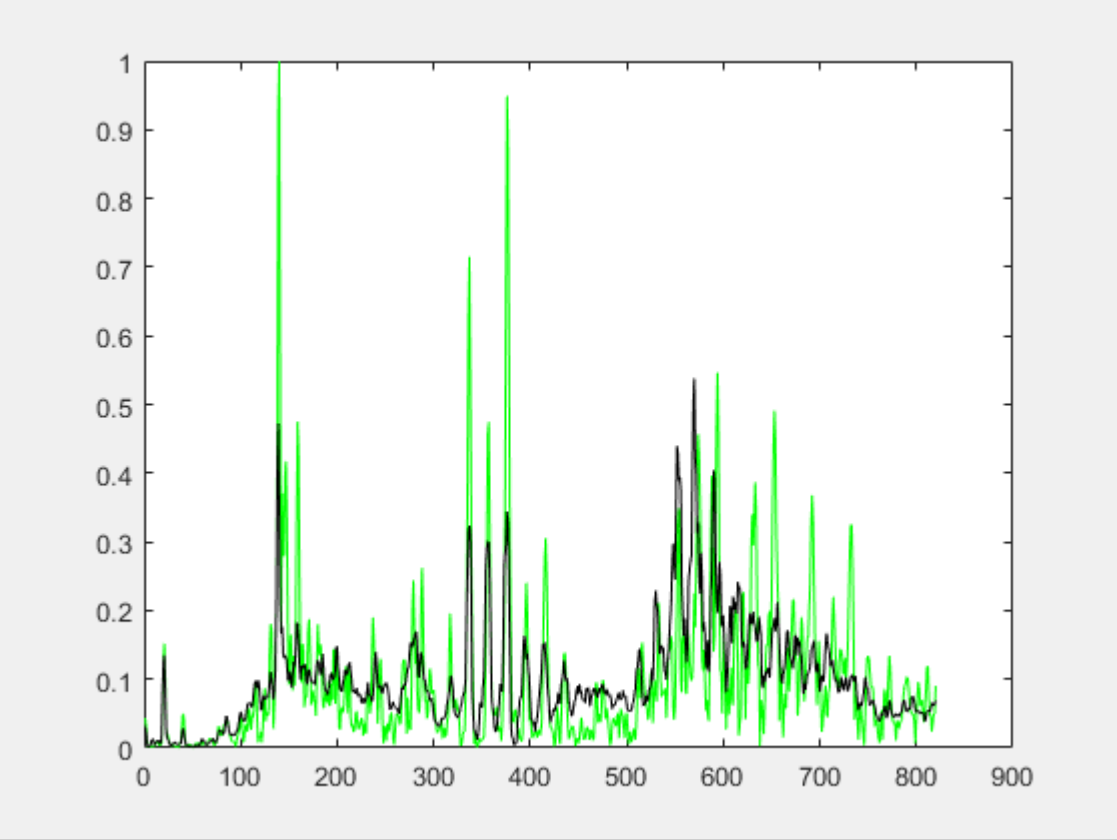


Figure 20 Healthy condition and reconstruction

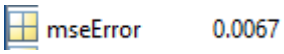


Figure 21 MSE for healthy condition

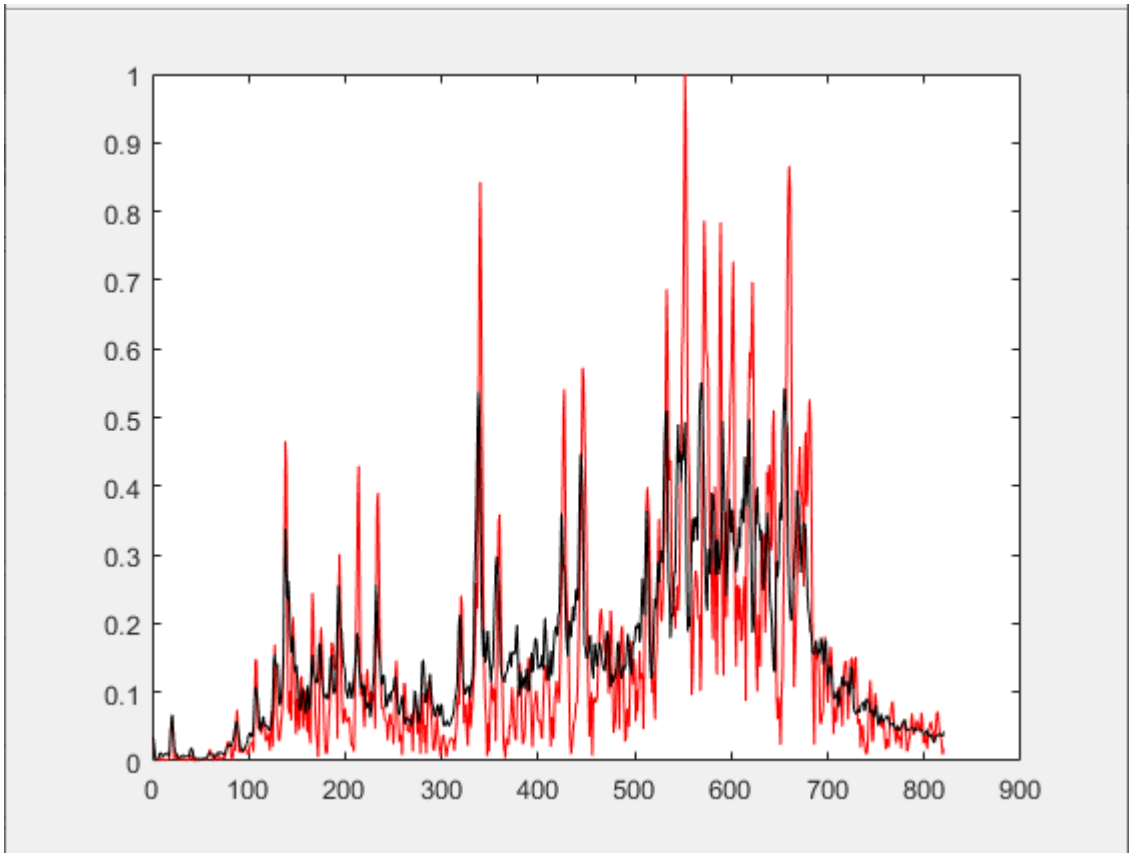


Figure 22 Bearings fault condition and reconstruction

mseError	0.0143

Figure 23 MSE for bearings fault

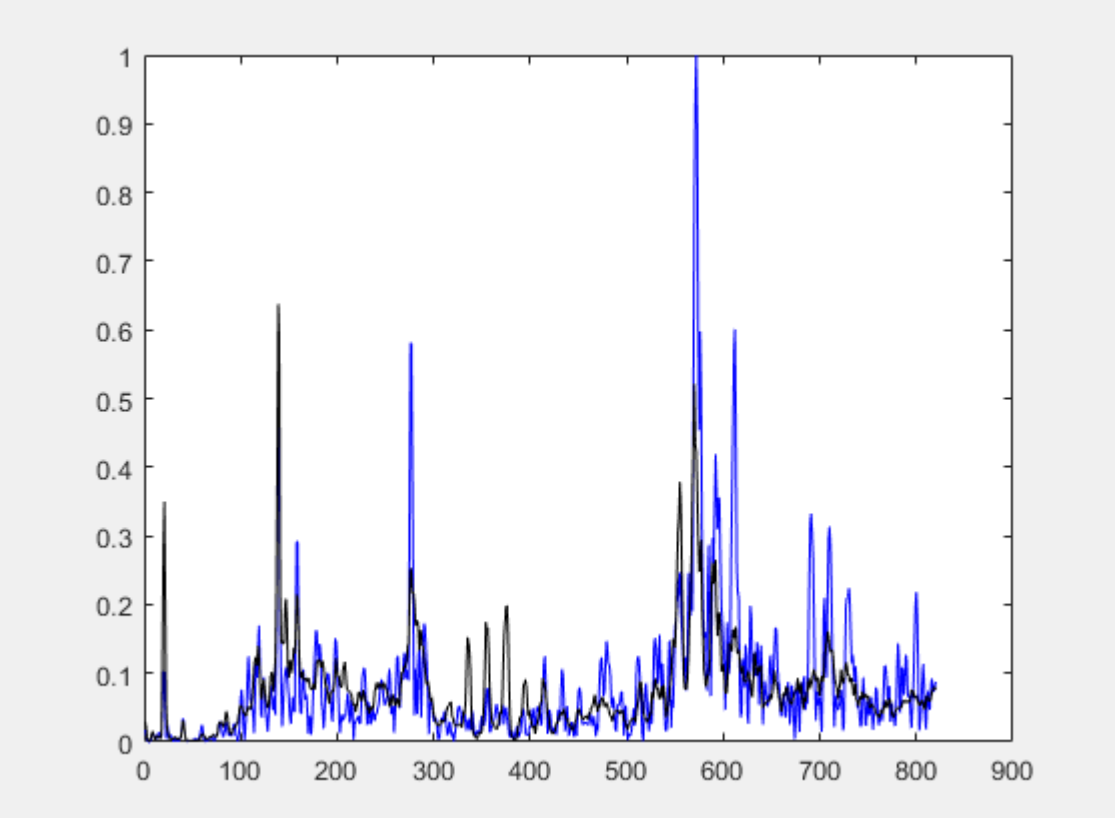


Figure 24 Demagnetization fault condition and its reconstruction

mseError	0.0052

Figure 25 MSE error for demagnetization fault

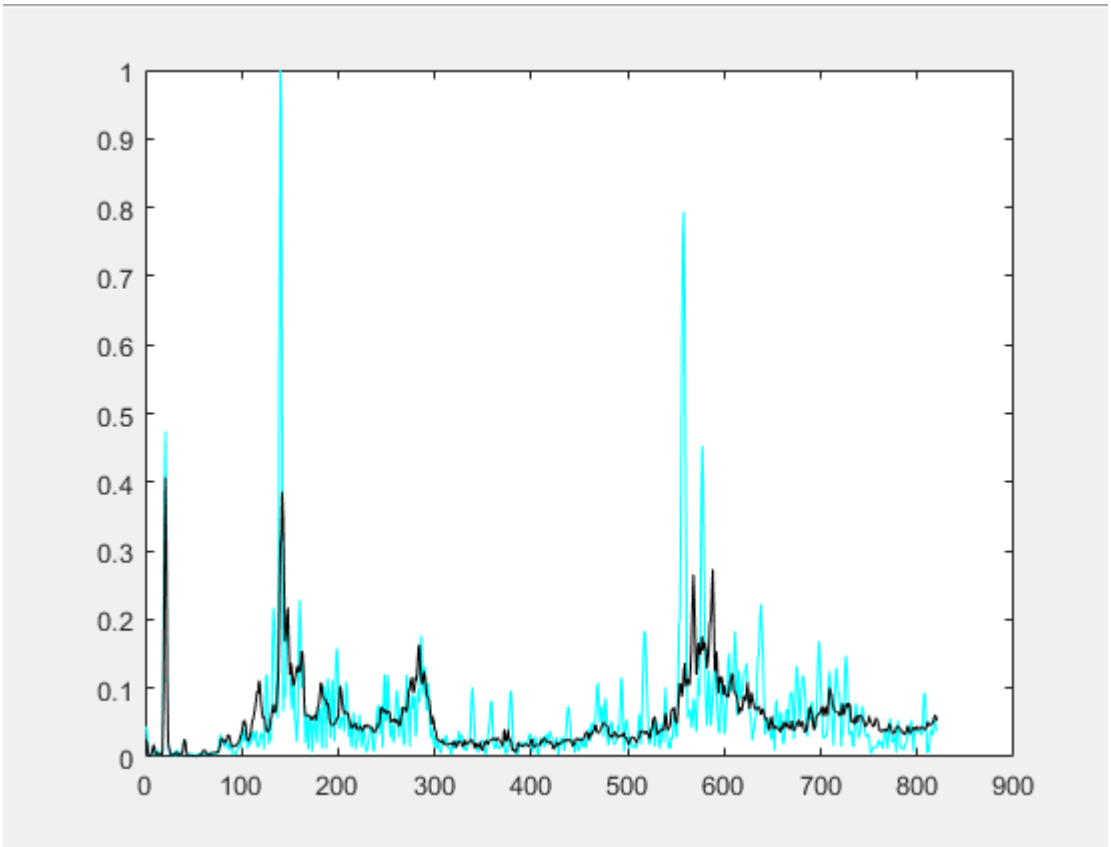


Figure 26 Eccentricity fault condition and reconstruction

mseError	0.0048

Figure 27 MSE error for eccentricity fault

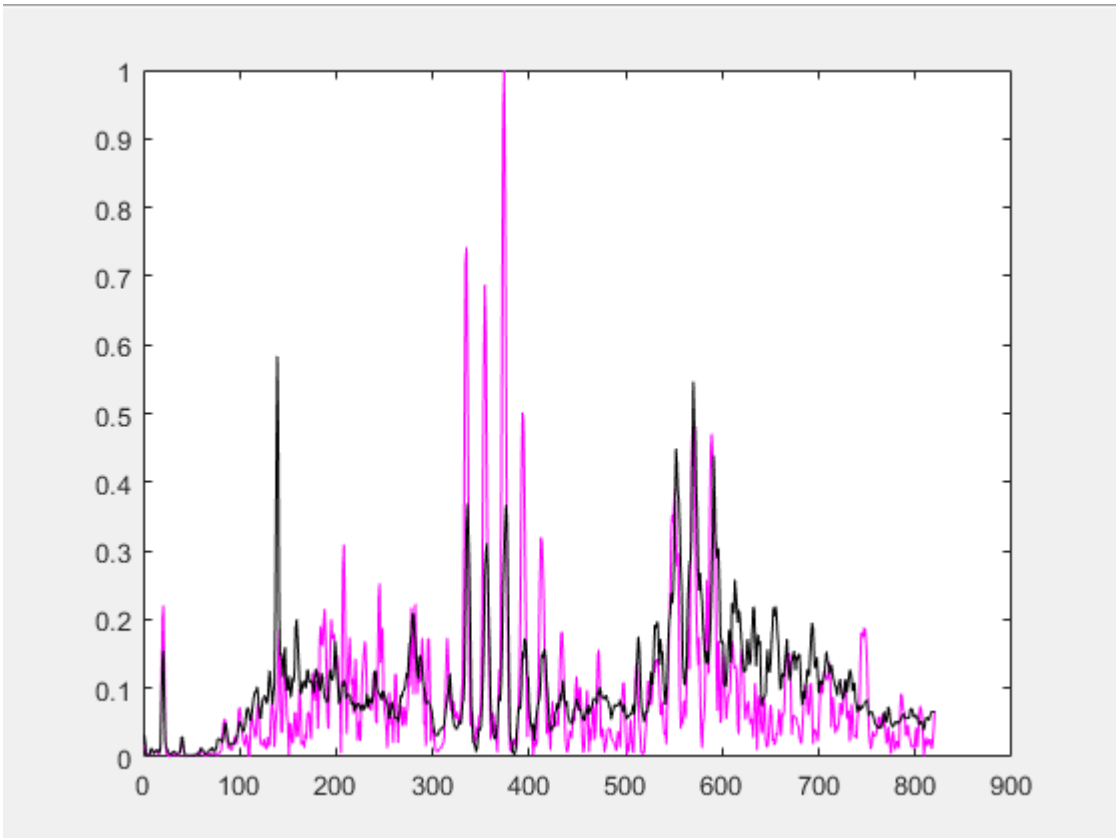


Figure 28 Gear fault condition and reconstruction

mseError	0.0085

Figure 29 MSE error for gear fault

Conclusions and recommendations for continuation of work

While working on this project many difirent concepts were studied, from gathering data from electromechanical machine to building and improving the model.

For continuation of the work I want to see how our model can be implemented in real world example. While this kind of tecnology is very important for future industry, especialy for quality of work.

Acknowledgements

I would like to dedicate special thanks to professor Dr. Miguel, for giving me a chance of working on this project, I learned a lot of the field of artificial intelligence and gained a lot of intrest of exploring the field also I would like to thank for his thoroughly guidance. My sincere thanks also goes Dr. Francisco for good explanation of harder topics and his time to meet with us.

Sources

- https://www.nature.com/articles/d41586-019-02212-4
- https://blogs.scientificamerican.com/sa-visual/unveiling-the-hidden-layers-of-deep-learning/
- http://expertsystem.com/machine-learning-definition/
- https://blogs.scientificamerican.com/sa-visual/unveiling-the-hidden-layers-of-deep-learning/
- http://dynapar.com/technology/vibration-analysis/
- https://developers.google.com/machine-learning/clustering/clustering-algorithms
- https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261
- https://www.intechopen.com/books/artificial-intelligence-emerging-trends-and-applications/artificial-intelligence-application-in-machine-condition-monitoring-and-fault-diagnosis

Works Cited

Ali, Y. H. (2018). *Artificial Intelligence Application in Machine Condition Monitoring and Fault Diagnosis.*

WASMER, K. (2019). *When AE (Acoustic Emission) meets AI.*